TAKE IT EASY: RELAXING CONTRASTIVE RANKING LOSS WITH CIDER

Technical Report

Theodore LAMORT DE GAIL, Dawid KICIŃSKI

Samsung R&D Institute Poland
Warsaw, Poland
t.lamort@samsung.com, d.kicinski@partner.samsung.com

ABSTRACT

This report presents our approach and results for task 6B of the DCASE2022 challenge concerning natural-language-based audio retrieval. To match the audio-text pairs, we learn cross-modal embeddings. The audio samples are encoded by an ensemble of four frozen expert models with transformer heads for time aggregation. Captions are encoded using a pre-trained language model. The model is trained with a modified contrastive ranking loss, enhanced with a heuristic caption similarity prior based on the CIDEr metric. We train the system on the AudioCaps and Clotho audio captioning datasets. Furthermore, we use a text classifier to gather additional useful audio-caption pairs from Freesound. We achieve 0.48 R@10 and 0.23 mAP@10 on the Clotho evaluation split (vs 0.19 and 0.07 respectively for the challenge baseline).

1. INTRODUCTION

The difficulty of this task, compared to its computer vision counterpart, is mainly due to the small amount of data available. In our approach, we propose to remedy this by (1) gathering additional data from Freesound in a semi-supervised fashion based on sentence classification (section 2), and (2) make heavy use of transfer learning, with four large-scale pre-trained audio expert models, as well as a language expert model.

We base our general framework on the work of Onescu et al. (2021) [1], which is the current state-of-the-art solution for text-to-audio retrieval on the Clotho dataset. We improve on their results by using a different aggregation strategy for the output of the audio experts: we apply transformers to the time series of embeddings, and replace the collaborative gating mechanism by unnormalised arithmetic mean, as detailed in section 3. In section 4, we explain how we also modify the loss function to (1) use CIDEr [2] similarity as a prior for caption embedding similarity, and (2) pull audio-caption embedding pairs closer together with a matching similarity loss component.

In section 5, we report our results on the evaluation split of Clotho v2, showing that they significantly outperform the DCASE2022 challenge baseline system. We also report our results on Clotho v1, showing that they outperform the solution in [1].

2. DATA

The training is performed on the provided Clotho dataset [3], as well as AudioCaps[4]. We experimented with the Hospital & Car dataset, where we translated the captions to English with DeepL, but we found that it does not improve results on the Clotho evaluation split. For AudioCaps, we use the standard train-validation split. For Clotho, we train on the development and validation subsets, and validate on the evaluation subset[1].

In an attempt to gather more data, we downloaded around 500K descriptions from Freesound, and trained a binary classifier to tell them apart from Clotho captions. The classifier was tasked to output 0 for Freesound and 1 for Clotho. The idea was that Freesound labels with outputs close to 1, i.e., the failure points of the classifier, would be useful audio captions similar to those in Clotho. We then downloaded all the samples corresponding to captions with an output score above 0.2, after filtering out those shorter than 3 seconds and longer than 2 minutes. This resulted in around 15 thousand additional training examples. Here is a sample of the captions found this way:

1. *Something is behind me and it's running towards me*
2. *The creaking and low rumbling of a ship or wooden space*
3. *Shower Water Running Drip*
4. *Flipping through the pages of a book quickly, multiple times*
5. *Steady engine noise in flight*
6. *A small amount of natural water running through a forest*
7. *Car passes on a country road, right to left making a whistling sound*
8. *A single thunder clap with rain*
9. *People talking and reverberation*
10. *Birds singing in the early morning + a car passing on a wet road*

The classifier was built by adding three fully connected layers of size 256 on top of the `all-mpnet-base-v2` model from SentenceTransformers -[5], with ReLU activation, and a single sigmoid-activated output neuron.

3. ARCHITECTURE

We train an audio encoder and a caption encoder with the same output dimension to learn cross-modal embeddings. At inference time, the matching audios are chosen using cosine similarity. For caption embeddings, we use SentenceTransformers[5], model `all-MiniLM-L6-v2`, which has an output dimension of 384. To encode audio, we first pre-compute per-time-frame embeddings from four expert models.

[1] According to Clotho naming, as opposed to DCASE Challenge naming.

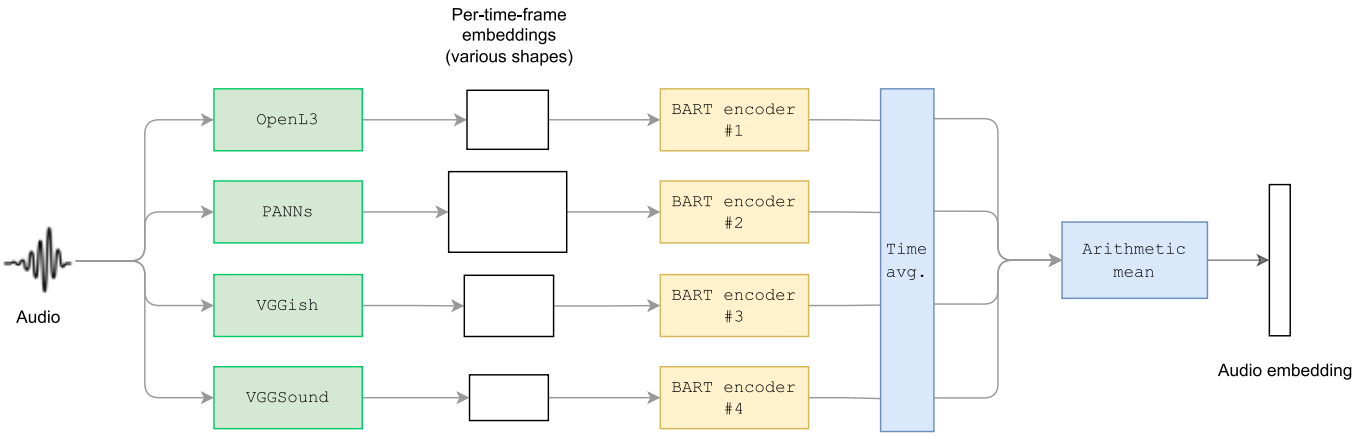

Figure 1: Audio encoder architecture

1. **OpenL3**[6, 7]. We use this[2] PyTorch implementation. The model has an inference window of 1 s. We use a hop length of 0.5 s. We choose `mel256` as input representation, `env` as content type and 512 as the embedding size. This results in approximately 2 embedding frames per second, and 512 embedding dimensions.
2. **PANNs**[8]. We employ the `Cnn14_16k` model available here[3]. In the original architecture, an embedding vector is constructed from the CNN output by first averaging over the frequency axis, and then taking the sum of the average and the maximum along the time axis. We remove the time aggregation operation, to get an output shape of about 3.1 embedding frames per second, and 2048 embedding dimensions.
3. **VGGSound**[9]. We use a ResNet-18 model pre-trained on the VGGSound dataset. It can be accessed here[4] (model H). We remove the VLADNet aggregation module, and aggregate the CNN output by averaging over the frequency axis. We found that the number of embedding frames per second produced this way was very high, so we added a one-dimensional average pooling layer along the time axis, with kernel size 60 and stride 40 to make the size more manageable. This results in about 2.3 frames per second and 512 embedding dimensions.
4. **VGGish**[10]. We use this[5] PyTorch implementation, without any modification, to produce one embedding frame per second, with dimension 128.

These time series of embeddings are sent through a single fully connected layer, and then fed into transformer networks. For each audio expert, we have a separate two-layer network with an internal dimension of 256, amounting to approx. 1.3M parameters each. These networks are based on the encoder from BART[11]. For each of the transformer outputs, we take the mean over the time axis to get a single embedding vector per audio expert, and we map it to the output dimension of the language model with another fully connected layer. Finally, these four vectors are averaged.

Note that the individual output vectors are not normalised prior to averaging, but the result is normalised when computing cosine similarity. Because of this, we can interpret the magnitude of the individual model outputs as confidence: the smaller the magnitude of a vector, the smaller the impact on the final result normalised to the hypersphere.

Since the audio experts embeddings are precomputed, these transformers and fully connected mapping layers constitute the trainable part of the audio encoder. Together with the caption encoder, which we set to be trainable, we have about 23M trainable parameters.

Table 1: Audio expert models used

	Parameters	Frames / sec	Dimension
OpenL3	4 687 042	2	512
PANNs	81 033 231	3.1	2048
VGGSound	11 328 757	2.3	512
VGGish	72 157 696	1	128

4. TRAINING

Loss. At first, we train the system using contrastive ranking loss, same as [1]. For a given batch, if we let $\alpha_1, \alpha_2, ..., \alpha_B$ be the audio embeddings and $\beta_1, \beta_2, ..., \beta_B$ be the corresponding caption embeddings, and we denote the cosine similarity between α_i and β_j by s_{ij} then the loss is expressed as

$$\mathcal{L} = \frac{1}{B} \sum_{1 \leq i,j \leq B,\ i \neq j} [m + s_{ij} - s_{ii}]_+ + [m + s_{ji} - s_{ii}]_+$$

where $[\cdot]_+$ is the hinge function $\max(\cdot, 0)$ and m is the *margin*, set to 0.2. We found that performance is very sensitive to batch size, and got best results with a batch size of 512.[6]

[2] github.com/torchopenl3/torchopenl3
[3] github.com/qiuqiangkong/audioset_tagging_cnn
[4] github.com/hche11/VGGSound
[5] github.com/harritaylor/torchvggish

[6] The training consumed around 15GB of VRAM, after using tricks such as training with 16-bit native precision. Please note that the footnote marker is not an exponent in this case.

When multiple captions are available per sample, we encode all of them, and replace the cosine similarity $s_{ij} = s(\alpha_i, \beta_j)$ with the average similarity to all caption embeddings, i.e.

$$s_{ij} = \frac{1}{n_{caps}} \left(s(\alpha_i, \beta_j^{(1)}) + s(\alpha_i, \beta_j^{(2)}) + ... + s(\alpha_i, \beta_j^{(n_{caps})}) \right).$$

While it is generally safe to assume that two captions describing two different samples should be embedded far apart, it could happen that there are similar captions describing different samples. If two such captions end up in the same batch, their embeddings will likely be close together and we will incorrectly punish the model with a high loss. For this reason, we turn to the CIDEr metric [2]. It gives us a crude measure of similarity, which we can use to refine our loss function. If c_{ij} is the CIDEr score between caption i and caption j, we modify the loss as follows:

$$\mathcal{L} = \frac{1}{B} \sum_{1 \leq i,j \leq B,\ i \neq j} [m \cdot r(c_{ij}) + s_{ij} - s_{ii}]_+ + [m \cdot r(c_{ij}) + s_{ji} - s_{ii}]_+$$

where r the "relax" function, which we still have to choose: if $r = 1$, we end up back with the original expression, if $r = 0$ the loss will be zero in most cases (very relaxed). We take the following approach: we find c_{80}, c_{90} and c_{99}, the 80th, 90th and 99th percentiles of all pairwise CIDEr scores in Clotho. We then define r to be piecewise linear such that:

1. $r(c) = 1$ for $c < c_{80}$
2. $r(c_{80}) = 1$
3. $r(c_{90}) = 0.7$
4. $r(c_{99}) = 0.4$
5. $r(c) = 0.4$ for $c > c_{99}$

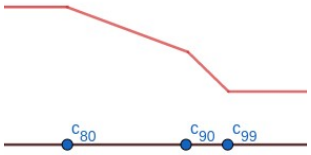

Figure 2: Relax function (not to scale)

This is obviously a very crude heuristic, and there is plenty of room for exploration and improvement here.

Additionally, we noticed that embeddings of matching audio-caption pairs had an average cosine similarity of around 0.6. In an ideal scenario, this should be 1 - we tried to increase this number by adding a "matching similarity" component to the loss, computed as $\mathcal{L}_M = -\frac{1}{B}(s_{11} + s_{22} + ... + s_{BB})$. We got best results when setting the loss to $\mathcal{L} + 3\mathcal{L}_M$.

Optimizer. We use AdamW with a learning rate of 1e-4.

Pre-training and fine-tuning. We tried pre-training on all combinations of Clotho, AudioCaps, and our home-made Freesound dataset. We got best results when pre-training on AudioCaps and Freesound but not Clotho, with training pairs being sampled from both datasets with equal frequency, and then fine-tuning on Clotho with the same hyperparameters.

Checkpointing and metrics. For saving checkpoints during training, we monitor validation loss, but also the following validation scores:

$$m_1 = \frac{1}{B} |\{i : s_{ii} = \max_j(s_{ij})\}|$$

$$m_2 = \frac{1}{B} |\{j : s_{jj} = \max_i(s_{ij})\}|$$

These correspond to the ranking accuracy versus non matching captions, and the ranking accuracy versus non matching audio. From each training, we keep 3 checkpoints per metric. Overall, all three of these metrics seem to correlate with evaluation mAP@10 metric quite well.

Length of training. We set epoch length to 16 steps. Our best model was pre-trained for 288 epochs and then finetuned for 8 epochs.

5. RESULTS

For the Clotho evaluation split (according to Clotho naming), we report the our results for each of our four submissions in Table 2. Please note that for competition purposes, this split was also used for validation and checkpointing.

Table 2: Evaluation scores on Clotho v2

	R@1	R@5	R@10	mAP@10
1.	0.137	0.354	0.484	0.231
2.	0.137	0.351	0.469	0.229
3.	0.137	0.354	0.470	0.228
4.	0.132	0.350	0.478	0.226

Additionally, to benchmark our system against Onescu et al. [1], we fine-tuned a separate model using only the development split of Clotho v1 (with a validation subset randomly selected as 20% of the development split), and evaluated it on the evaluation split of Clotho v1. We show the results in Table 3. The results reported in the aforementioned paper, with pre-training on AudioCaps only, were 9.6 ± 0.3 R@1 and 40.1 ± 0.7 R@10.

Table 3: Evaluation scores on Clotho v1

	R@1	R@5	R@10	mAP@10
1.	0.120	0.344	0.478	0.214

6. REFERENCES

[1] A.-M. Oncescu, A. S. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries," 2021. [Online]. Available: https://arxiv.org/abs/2105.02192

[2] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," 2014. [Online]. Available: https://arxiv.org/abs/1411.5726

[3] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," 2019. [Online]. Available: https://arxiv.org/abs/1910.09387

[4] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *NAACL-HLT*, 2019.

[5] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[6] R. Arandjelovic and A. Zisserman, "Look, listen and learn," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[7] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, p. 2880–2894, 2020.

[9] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[10] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," 2016. [Online]. Available: https://arxiv.org/abs/1609.09430

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1910.13461